AMENDMENTS TO THE CLAIMS

Claims 1-105 were filed originally.

Claims 1-21, 35-55, 69-80, and 91-100 are canceled.

No claims are amended.

Accordingly, claims 22-34, 56-68, 81-90, and 101-105 remain pending.

Claims 1-21 **(Canceled)**.

22. **(Original)** A method comprising:

segmenting a file into multiple blocks;

computing hashes of each of the blocks to produce corresponding block hash values;

encrypting the blocks using their corresponding block hash values as encryption keys to produce encrypted blocks;

storing the encrypted blocks as a primary data stream;

creating an indexing structure to index individual encrypted blocks, the indexing structure containing a leaf node for each corresponding encrypted block, the leaf node containing an access value formed by encrypting the block hash value for the corresponding encrypted block using an access key and a verification value formed by hashing the corresponding encrypted block;

storing the indexing structure in a separate metadata stream; and

encrypting the access key using a public key of a user who is granted access to the file.

23. **(Original)** A method as recited in claim 22, wherein the segmenting comprises dividing the file into equal size blocks.

24. **(Original)** A method as recited in claim 22, wherein the encrypting of the blocks comprises encrypting each block using a symmetric cryptographic cipher and the corresponding block hash value as the symmetric encryption key.

25. **(Original)** A method as recited in claim 22, further comprising verifying an authenticity of a target encrypted block independently of other encrypted blocks by traversing the indexing structure to a leaf node associated with the target encrypted block and using the verification value in the leaf node associated with the target encrypted block.

26. **(Original)** A method as recited in claim 22, further comprising:

traversing the indexing structure to a leaf node associated with a target block;

decrypting the target block using the access value of the leaf node associated with the target block; and

reading the target block following said decrypting.

27. **(Original)** A method as recited in claim 26, further comprising:

modifying the target block of the file to produce a modified target block;

computing a hash value of the modified target block;

encrypting the modified target block using the hash value as an encryption key to produce a modified encrypted block; and

recreating a new leaf node for the modified encrypted block.

28. **(Original)** A method as recited in claim 22, wherein the creating further comprises:

grouping leaf nodes into multiple groups;

hashing each group of leaf nodes to form intermediate nodes of the indexing structure; and

hashing an array of the intermediate nodes to produce a root.

29. **(Original)** A method as recited in claim 28, wherein the constructing further comprises digitally signing at least the root.

30. **(Original)** A method as recited in claim 22, further comprising digitally signing at least a portion of the metadata stream.

31. **(Original)** A method as recited in claim 22, further comprising generating a delegation certificate that grants other entities permission to collectively authenticate the file in absence of the signature of a last writer to the file.

32. **(Original)** A method as recited in claim 22, wherein the file comprises a sparse file in which at least one of the blocks contains no data, the method further comprising:

differentiating non-data blocks of the sparse file that contain no substantive content from the data blocks of the sparse file that contain substantive data; and

deallocating portions of the metadata stream that pertain to the non-data blocks in the data stream.

33. **(Original)** A data structure, embodied on a computer-readable medium, produced by the method of claim 22.

34. **(Original)** One or more computer readable media comprising computer-executable instructions that, when executed, perform the method as recited in claim 22.

Claims 35-55 **(Canceled)**.

56. **(Original)** One or more computer readable media comprising computer-executable instructions that, when executed, direct a computing device to:

segment a file into multiple blocks;

hash each of the blocks to produce block hash values;

encrypt the blocks using their corresponding block hash values as encryption keys to produce encrypted blocks;

create an indexing structure to index individual encrypted blocks, the indexing structure containing a leaf node for each corresponding encrypted block, the leaf node containing an access value formed by encrypting the block hash value for the corresponding encrypted block using an access key and a verification value formed by hashing the corresponding encrypted block;

encrypt the access key using a public key of a user who is granted access to the file.

57. **(Original)** One or more computer readable media as recited in claim 56, further comprising computer-executable instructions that, when executed, direct a computing device to:

store the encrypted blocks as a primary data stream; and

store the indexing structure in a separate metadata stream.

58. **(Original)** One or more computer readable media as recited in claim 56, further comprising computer-executable instructions that, when executed, direct a computing device to segment the file into equal size blocks.

59. **(Original)** One or more computer readable media as recited in claim 56, wherein the blocks are encrypted using a symmetric cryptographic cipher and the access key is encrypted using an asymmetric cryptographic cipher.

60. **(Original)** One or more computer readable media as recited in claim 56, further comprising computer-executable instructions that, when executed, direct a computing device to verify an authenticity of a target encrypted block

independently of other encrypted blocks by traversing the indexing structure to a leaf node associated with the target encrypted block and using the verification value in the leaf node associated with the target encrypted block.

61. **(Original)** A method as recited in claim 60, wherein the indexing structure contains a root and zero or more intervening nodes between the root and the leaf nodes, the traversing further comprising verifying an authenticity of the root and any intervening nodes on a path from the root to the leaf node associated with the target encrypted block.

62. **(Original)** One or more computer readable media as recited in claim 56, further comprising computer-executable instructions that, when executed, direct a computing device to:

decrypt a target block using an access value of a leaf node associated with the target block; and

read the target block after it is decrypted.

63. **(Original)** One or more computer readable media as recited in claim 62, further comprising computer-executable instructions that, when executed, direct a computing device to:

modify the target block to produce a modified target block;

hash the modified target block to produce a hash value;

encrypt the modified target block using the hash value as an encryption key to produce a modified encrypted block; and

recreate a new leaf node for the modified encrypted block.

64. **(Original)** One or more computer readable media as recited in claim 56, further comprising computer-executable instructions that, when executed, direct a computing device to:

group leaf nodes into multiple groups;

hash each group of leaf nodes to form intermediate nodes of the indexing structure; and

hash an array of the intermediate nodes to produce a root.

65. **(Original)** One or more computer readable media as recited in claim 64, further comprising computer-executable instructions that, when executed, direct a computing device to digitally sign at least the root.

66. **(Original)** One or more computer readable media as recited in claim 56, further comprising computer-executable instructions that, when executed, direct a computing device to digitally sign at least a portion of the metadata stream.

67. **(Original)** One or more computer readable media as recited in claim 56, further comprising computer-executable instructions that, when executed, direct a computing device to generate a delegation certificate that grants other entities permission to collectively authenticate the file in absence of the signature of a last writer to the file.

68. **(Original)** One or more computer readable media as recited in claim 56, wherein the file comprises a sparse file in which at least one of the blocks contains no substantive data, the media further comprising computer-executable instructions that, when executed, direct a computing device to:

differentiate non-data blocks of the sparse file that contain no substantive content from the data blocks of the sparse file that contain substantive data; and

deallocate portions of the metadata stream that pertain to the non-data blocks in the data stream.

Claims 69-80 **(Canceled)**.

81. **(Original)** A component in a distributed file system in which file are stored across multiple distributed computers, the component comprising:

a segmenting module to divide a file into multiple blocks;

a hash module to hash each of the blocks to produce block hash values;

a cryptographic engine to encrypt the blocks using their corresponding block hash values as encryption keys to produce encrypted blocks; and

an index builder to create an indexing structure for indexing individual encrypted blocks, the indexing structure containing a leaf node for each corresponding encrypted block, the leaf node containing an access value formed by encrypting the block hash value for the corresponding encrypted block using an

access key and a verification value formed by hashing the corresponding encrypted block.

82. **(Original)** A component as recited in claim 81, wherein the cryptographic engine is further configured to encrypt the access key using a key of a user who is granted access to the file.

83. **(Original)** A component as recited in claim 81, wherein the segmenting module divides the file into equal size blocks.

84. **(Original)** A component as recited in claim 81, wherein cryptographic engine employs a symmetric cryptographic cipher to encrypt the blocks.

85. **(Original)** A component as recited in claim 81, further comprising a verification module to verify an authenticity of a target encrypted block independently of other encrypted blocks by traversing the indexing structure to a leaf node associated with the target encrypted block and using the verification value in the leaf node associated with the target encrypted block.

86. **(Original)** A component as recited in claim 85, wherein the indexing structure contains a root and zero or more intervening nodes between the root and the leaf nodes, the verification module being configured to verify an authenticity of the root and any intervening nodes on a path from the root to the leaf node associated with the target encrypted block.

87. **(Original)** A component as recited in claim 81, further comprising a control module to index into the indexing structure to a leaf node associated with a target block, decrypt the target block using the access value of the leaf node associated with the target block, and read the target block.

88. **(Original)** A component as recited in claim 87, where upon modification of the target block:

the hash module hashes the modified target block to produce a new hash value;

the cryptographic engine encrypts the modified target block using the new hash value as an encryption key to produce a modified encrypted block; and

the index builder creates a new leaf node for the modified encrypted block.

89. **(Original)** A component as recited in claim 81, wherein the index builder is configured to create intermediate nodes that index the leaf nodes.

90. **(Original)** A component as recited in claim 81, further comprising a signing module to digitally sign at least a portion of the indexing structure.

Claims 91-100 **(Canceled)**.

101. **(Original)** A data structure stored on a computer-readable medium, comprising:

multiple encrypted file blocks, each encrypted file block being encrypted by a symmetric cipher that uses a hash of the block as an encryption key; and

an indexing structure to index individual encrypted file blocks independently of other encrypted file blocks.

102. **(Original)** A data structure as recited in claim 101, wherein the indexing structure comprises a leaf node for each corresponding encrypted block, the leaf node containing an access value formed by encrypting the hash of the block using a randomly generated key and a verification value formed by hashing the corresponding encrypted block.

103. **(Original)** A data structure as recited in claim 102, further comprising a user key list containing one or more identities of user who have access to the encrypted file blocks, each identity including an entry with an encrypted version of the randomly generated key that is encrypted using the user's public key.

104. **(Original)** A data structure as recited in claim 101, wherein the indexing structure comprises:

a leaf node for each corresponding encrypted block, the leaf node containing an access value formed by encrypting the hash of the block using a

randomly generated key and a verification value formed by hashing the corresponding encrypted block; and

a root node formed by hashing an array of the leaf nodes.

105. **(Original)** A data structure as recited in claim 104, wherein the indexing structure further comprises a digital signature produced by digitally signing at least the root node.